

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2002年 9月17日

出 願 番 号

Application Number:

特願2002-270324

[ST.10/C]:

[JP2002-270324]

出 願 人

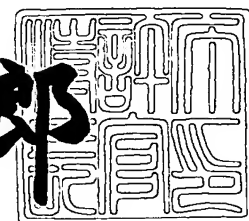
Applicant(s):

パイオニア株式会社

2003年 6月26日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2003-3050532

【書類名】 特許願

【整理番号】 57P0127

【提出日】 平成14年 9月17日

【あて先】 特許庁長官 殿

【国際特許分類】 H04L 1/00
H04L 13/00
H03M 13/00

【発明者】

【住所又は居所】 東京都大田区大森西4丁目15番5号 パイオニア株式会社 大森工場内

【氏名】 勝屋 宏一

【特許出願人】

【識別番号】 000005016

【氏名又は名称】 パイオニア株式会社

【代理人】

【識別番号】 100083839

【弁理士】

【氏名又は名称】 石川 泰男

【電話番号】 03-5443-8461

【手数料の表示】

【予納台帳番号】 007191

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9102133

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 フレーム構造のノイズ除去装置、フレーム構造のノイズ除去方法およびフレーム構造のノイズ除去プログラム

【特許請求の範囲】

【請求項 1】 符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、

前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする、ノイズ除去装置。

【請求項 2】 請求項 1 に記載のノイズ除去装置において、

前記エラー検出手段は、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、ノイズ除去装置。

【請求項 3】 請求項 1 または 2 に記載のノイズ除去装置において、

前記エラー検出手段は、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定することを特徴とする、ノイズ除去装置。

【請求項 4】 請求項 1 から 3 のいずれかに記載のノイズ除去装置において

前記デコード手段は、前記エラー検出時にデコード結果が 0 となる前記符号化された音声データをデコードすることを特徴とする、ノイズ除去装置。

【請求項 5】 符号化された音声データにエラーが発生したかどうかを検出するエラー検出ステップと、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコードステップを備え、

前記デコードステップは、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする、ノイズ除去方法。

【請求項 6】 請求項 5 に記載のノイズ除去方法において、

前記エラー検出ステップは、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、ノイズ除去方法。

【請求項 7】 請求項 5 または 6 に記載のノイズ除去方法において、

前記エラー検出ステップは、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定するステップであることを特徴とする、ノイズ除去方法。

【請求項 8】 請求項 5 から 7 のいずれかに記載のノイズ除去方法において

前記デコードステップは、前記エラー検出時にデコード結果が 0 となる前記符号化された音声データをデコードするステップであることを特徴とする、ノイズ除去方法。

【請求項 9】 コンピュータに、符号化された音声データにエラーが発生したかどうかを検出するエラー検出機能と、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード機能を備えさせ、

前記デコード機能は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを実現させるためのプログラム。

【請求項 1 0】 請求項 9 に記載のプログラムにおいて、

前記エラー検出機能は、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、プログラム。

【請求項 1 1】 請求項 9 または 1 0 に記載のプログラムにおいて、

前記エラー検出機能は、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定する機能であることを特徴とする、プログラム。

【請求項 1 2】 請求項 9 から 1 1 のいずれかに記載のプログラムにおいて

前記デコード機能は、前記エラー検出時にデコード結果が 0 となる前記符号化された音声データをデコードすることを特徴とする、プログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、デジタルオーディオデータフレーム、圧縮音声データフレームその他のフレーム構造のノイズ除去に関する。

【0002】

【従来の技術】

【特許文献1】 特開平8-286698

【特許文献2】 特開2000-59231

【特許文献3】 特表2001-501063

【特許文献4】 特開2002-73091

従来の圧縮音声再生方法においては、圧縮音声ストリーム中のフレームエラーを検出する方法としては、CRC (Cyclic Redundancy Check) によるエラーチェックが一般的である。圧縮音声再生方法の一つであるAAC (Advanced Audio Coding) の場合、CRCでエラー検出対象としている範囲は、圧縮音声ストリームの全体ではなく、一部分である。圧縮音声ストリームはフレームの集合からなるが、ISDB-TSB (Integrated Services Digital Broadcasting-Terrestrial Sound Broadcasting) : (国内デジタル地上波オーディオ放送) ではAACのなかのADTS (Audio Data Transport Stream) フレームを使用している。図1はADTSフレームの構造を示す。ADTSフレームはADTSヘッダー、CRC、Raw_data_blockの3つの部分からなる。ADTSヘッダーにはフレームの各種情報が書き込まれている。CRCにはADTSフレームの一部分についてエラーチェックをした結果が書き込まれている。Raw_data_blockには圧縮音声データおよび圧縮音声データの種別を示す情報が書き込まれている。図2はRaw_data_blockの構造を示す。Raw_data_blockは、圧縮音声データの種別を示すID、圧縮音声データであるSyntactic Element、その他データであるbyte alignmentから構成されている。ISDB-TSBのチャンネルの構成やプ

ロファイルにより、IDの種類や数が異なる。図3にIDの種類を示す。

【0003】

図3には左欄から、Syntactic Elementの名称、IDの名称、IDのコード、略語の順に8種類が記載されている。上から3番目のID_CCEはISDB-TSBでは使用されない。

【0004】

図4に従来のAACデコーダのエラー検出方法のフローチャートを示す。ステップS1では音声圧縮ストリーム中の1フレームが入力される。ステップS2では、ステップS1において入力された1フレーム中のヘッダー部分を取得する。ヘッダー部分には、ID、レイヤ、保護ビット、プロファイル、およびサンプリング周波数等の情報が格納されている。ステップS3では、ステップS2で取得したヘッダー情報がAACの仕様に一致しているかいないかを判断する。ヘッダー情報がAACの仕様に一致している場合にはステップS4へ進み、一致していない場合にはステップS8に進む。

【0005】

ステップS4では、ADTSフレームのうちRaw_Data_Block部分を取得する。ここには図2で説明した各種IDとSyntactic Elementが格納されている。図3に表示したIDと同一のものが含まれる場合にはステップS6に進み、図3に表示したIDと異なるものが含まれる場合には、ステップS8に進む。すなわち、各IDの種類を示すコードが0X0～0X7に該当する場合には、ステップS6に進み、コードが0X0～0X7に該当しない場合には、ステップS8に進む。

【0006】

ステップS6ではCRCチェックを行う。AACのADTSフレームにおいてCRCチェックの対象となるのは、ADTSヘッダーの全てと、図2のIDのうちSCE、CPE、CCE、LFEの最初の192ビットと、CPE内にある2番目のchannel_pair_elementの最初128ビットと、PCE、DSEの全てのデータである。これらのCRCのチェック対象部分を生成多項式にいて計算し、ADTSヘッダー直後のCRCの値と比較して、値が一致

すればフレーム内にエラーがなかったとしてステップ S 7 に進み、値が一致しなければエラーがフレーム内にあるものとしてステップ S 8 に進む。従って、各 I D、fill__element、および S C E、C P E、C C E、L F E の 1 9 3 ビット以降、C P E 内にある 2 番目の channel__pair__element の最初から 1 2 9 ビット以降の部分はエラーチェックの対象とはなっていない。

【 0 0 0 7 】

ステップ S 7 ではフレーム内にエラーが発生していない A A C フレームについてはデコード処理を行う。

【 0 0 0 8 】

ステップ S 8 では、フレーム内でエラーが発生したと判断された場合の処理を行う。フレーム内でエラーが発生したと判断された場合には、デコード処理を行わずにデコード結果のデータを全て “ 0 ” で置き換える。

【 0 0 0 9 】

この場合に、エラーが発生したフレームは、“ 0 ” として出力し、ソフトミュート（フェードイン／フェードアウト）を施すことにより、エラー発生時における聞きにくさを解消しようとする（第 1 の従来技術）。その他にも、エラー発生時における聞きにくさを解消するために前のデータを繰り返すなどの方法（第 2 の従来技術）が採用されている。

【 0 0 1 0 】

図 5 および図 6 を用いてこれらの場合について説明する。図 5 は第 1 の従来技術を示す概略図であり、図 6 は第 2 の従来技術を示す概略図である。図 5 の上段は、入力データ、中段は、入力データのデコーダ、下段は出力データを示す。入力はフレーム毎に C R C が付加されており、図 1 ではフレーム 0、フレーム 1、フレーム 2 の 3 つが図示されている。フレーム 1 の C R C においてエラーが発見された場合、フレーム 1 をデコードした結果のフレーム 1 A のデータは、すべて “ 0 ” とされる。その結果、フレーム 0 A とフレーム 2 A 間でデータが途切れるので、フレーム 0 A のフレーム 1 A に近い部分の出力を徐々に小さくしていき（フェードアウト）、フレーム 2 A のフレーム 1 A に近い部分を小さい出力から徐

々に大きくしていく（フェードイン）。このようにして、エラー発生時における聞きにくさを解消する方法がとられる。

【 0 0 1 1 】

図 6 の上段は、入力データ、中段は、入力データのデコーダ、下段は出力データを示す。入力はフレーム毎に C R C が付加されており、図 6 ではフレーム 3、フレーム 4、フレーム 5 の 3 つが図示されている。フレーム 4 の C R C においてエラーが発見された場合、フレーム 4 をデコードした結果のフレーム 4 A のデータは、エラーが発生していない直前のフレーム 3 A のデータにすべて置き換えられる。このようにして、エラー発生時における聞きにくさを解消する方法がとられる。

【 0 0 1 2 】

【発明が解決しようとする課題】

図 4 に示す従来技術のエラー検出方法においては、C R C でエラー検出ができるデータはフレームの一部のみであるので、エラー検出能力が低いという問題点を有していた。また、図 5 に示すソフトミュート（フェードイン／フェードアウト）によるエラー処理を行っても、出力音が有音（正常フレーム）から無音（エラー発生フレーム）に変化し、更に有音（正常フレーム）に変化することになるので、聴きづらいと感ずる場合もあった。また、図 6 において説明した、エラー検出前のフレームのデータを使用する場合にも、エラー検出フレームの前のフレーム音が間延びしたような感覚や、エラー検出フレームの直後のフレーム音が音飛びしたような感覚を生じる場合があった。

【 0 0 1 3 】

そこで、本発明では、C R C によるエラー検出よりもさらにエラー検出能力を高め、かつ、エラー検出時における出力音をより聞きやすい音に改善することを課題の一例としている。

【 0 0 1 4 】

【課題を解決するための手段】

上記の課題を解決するために、請求項 1 に記載の発明は、符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、前記符号化され

た音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする。

【 0 0 1 5 】

【発明の実施の形態】

－第 1 の実施形態－

以下、ノイズ除去装置による第 1 の実施形態について説明する。図 7 は、本実施形態のエラー検出方法をフローチャートで示したものである。

【 0 0 1 6 】

本実施形態では I S D B - T S B (国内デジタル地上波オーディオ放送) において使用されている A A C における A D T S フレームの処理について説明を行う。

【 0 0 1 7 】

ステップ S 9 ではデータストリーム中の 1 フレームが入力される。ステップ S 1 0 では、ステップ S 9 において入力された 1 フレーム中のヘッダー部分を取得する。ヘッダー部分には、I D、レイヤ、保護ビット、ビットレート、およびサンプリング周波数等の情報が格納されている。ステップ S 1 1 では、ステップ S 1 0 で取得したヘッダー情報が I S D B - T S B の仕様を決めている A R I B (Association of Radio Industries and Businesses) の仕様に一致しているかいないかを判断する。ヘッダー情報が A R I B の仕様に一致している場合にはステップ S 1 2 へ進み、一致していない場合にはステップ S 1 7 に進む。たとえば、サンプリング周波数について、A A C の規格では 1 2 種類が定義されているが、I S D B - T S B の A R I B ではそのうち 4 8 k H z、3 2 k H z、2 4 k H z の 3 種類のみしか使用していないので、サンプリング周波数が 4 8 k H z、3 2 k H z、2 4 k H z の場合にはステップ S 1 2 へ進み、サンプリング周波数が 4 8 k H z、3 2 k H z、2 4 k H z でない場合にはステップ S 1 7 へ進むことになる。

【 0 0 1 8 】

ステップ S 1 2 では、A D T S フレームのうち R a w _ D a t a _ B l o c k

部分を取得する。ここには図3で説明した各種IDとSyntactic Elementが格納されている。ステップS13では各種IDのうちISDB-TSBで使用していないIDをRaw_Data_Block部分にもつ場合にはS9に進み、ISDB-TSBで使用しているIDをRaw_Data_Block部分にもつ場合にはステップS14に進む。

【0019】

ステップS14では、CRC (Cyclic Redundancy Check) によってエラーが発生しているかどうかをチェックする。このCRCチェックは従来から実施されていたものである。CRCによってエラーが検出された場合にはステップS17に進み、CRCによってエラーが検出されなかった場合にはステップS15に進む。

【0020】

ステップS15では、ADTSヘッダーに書き込まれているフレーム長さ情報に基づいて、処理したフレーム全体の長さがフレーム長さ情報どおりか否かを判断している。すなわち、ヘッダーの先頭から処理したビット数をカウントし、図2に示すbyte alignmentの最後のビットまで計算する。そのビット数がADTSヘッダーに書き込まれているフレーム長さ情報と一致するか否かを判断する。一致しない場合は、エラーがあるものとみなして、ステップS17に進み、一致する場合は、エラーがないものとみなして、ステップS16に進む。ステップS16に進む場合には、取得しているフレームの内容をメモリに書き込む。

【0021】

ステップS16では、取得したフレームをデコード処理する。この動作について図8を参照して説明する。図8の上半分はエンコード処理を表わし、下半分はここで説明するデコード処理を表わす。周波数サンプル列9内のフレーム21、22、23がステップS9で取得する各フレームに対応する。デコード処理の最初は、各フレームに逆変形離散コサイン変換IMDCT (Inverse Modified Discrete Cosine Transform) 10処理を行う。IMDCT10の変換式は以下のように表わされる。

【 0 0 2 2 】

【数 1】

$$X_{i,n} = 2/N \sum_{k=0}^{N/2-1} \text{spec}[i][k] \cos(2\pi/N(n+n_0)(k+1/2))$$

$0 \leq n < N$ 、(n: sample index, i: windows index, k: spectral coefficient index, N: window length based on the window sequence value, $n_0 = (N/2 + 1)/2$)

次に、IMDCT10の出力ブロック21A、22A、23Aに窓関数11をかける。この窓関数11は一種のフィルタとして考えることもできる。各フレームは周波数特性をもつが、窓関数11によりこの周波数特性が決定される。また、窓関数11を用いることにより前後のブロックの連続性を確保する。AACでは、窓関数11としてサイン窓と隣接バンドとの分離度の良いガイザー窓の2種類が定義されており、本実施形態ではいずれの窓関数をも選択することが可能である。

【 0 0 2 3 】

窓関数11の処理は一のIMDCT10の出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバーラップさせて行う。図8においてはブロック22Aはその隣接するブロック21Aとブロック23Aのブロックサイズの半分ずつをオーバーラップさせて窓関数11の処理を行っている。その後、窓関数11の処理を行った出力ブロックの隣接するブロックのブロックサイズの半分ずつを加算12して、信号を再生し、時間サンプル列13を生成する。

【 0 0 2 4 】

したがって、フレーム22にエラーがあっても、デコード結果が0となっても、フレーム22のデコード結果が一部となる対応する時間サンプル列ブロック25および26には、ブロック21Aの後半部と、ブロック23Aの前半部がデータとして入るので、無音となることはなく、しかも前後のブロック（ブロック21とブロック23）からのデコード処理結果がデータとして入るので、ブロック22と関連した連続性を確保したデータが入ることになる。

【 0 0 2 5 】

ステップS17では、ステップS9からステップS15までの間のエラーチェ

ックにおいて、フレーム内にエラーが発生していないと判断されたフレームが記憶される。したがって、フレーム内にエラーが発生した場合に、ステップ S 1 6 にてデコード処理が行われるフレームは、ステップ S 1 7 に記憶されている、エラーが発生していない直前のフレームとなる。フレームにエラーが発生した場合でもステップ S 1 6 のデコード処理について説明したように、窓関数 1 1 の処理は前後のフレームのデータを使用しているため、前後のデータの連続性は確保されており、再生された信号の音が突然途切れることはない。

【 0 0 2 6 】

AAC はブロック符号化を行っていることから、デコーダで時間信号に戻したときに、圧縮歪みはブロック内に歪みとして広がる。ブロックごとに圧縮の仕方が異なれば、時間信号にもどしたときにブロック間で不連続が発生し、ブロック歪みとよばれる歪みが発生する。オーディオの場合、この非連続なブロック歪みの音は人間に不快感を与えやすい。このため、ステップ S 1 7 において、オーバーラップして窓関数 1 1 処理をして前後のデータの連続性を確保して滑らかにブロック間をつなぐことにより、ブロック歪みが軽減されることになる。

－ 第 2 の実施形態 －

以下、ノイズ除去装置による第 2 の実施形態について説明する。

【 0 0 2 7 】

図 9 は、本実施形態のエラー検出方法をフローチャートで示したものである。第 1 の実施形態と重複する部分は同一符号を付した。

【 0 0 2 8 】

第 2 の実施形態は、第 1 の実施形態とステップ S 1 8 における処理の内容が異なる。ステップ S 1 8 において記憶されるフレームデータはステップ S 1 6 のデコード処理の結果が 0 となるフレームデータである。すなわち、図 9 中ステップ S 1 1、S 1 3、S 1 4、S 1 5 においてフレームにエラーがあると判断された場合には、当該フレームのデコード結果は 0 になる。しかし、ステップ S 1 6 における窓関数処理は一の IMDCT の出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバーラップさせた処理を行う。すなわち、図 8 において、ブロック 2 2 A はその隣接するブロック 2 1 A とブロック 2 3 A のブ

ロックサイズの半分づつをオーバーラップさせて窓関数処理を行う。そして、窓関数処理を行った出力ブロックの隣接するブロックのブロックサイズの半分づつを加算して出力信号を再生する。したがって、当該フレーム出力結果が0であっても、当該フレームの前後のフレームをデコードした結果が当該フレームのデコード結果としての出力信号になる。すなわち、デコードされた信号は、従来技術のように有音（正常フレーム）から無音（エラー発生フレーム）へ、無音（エラー発生フレーム）から有音（正常フレーム）へと変化するのではなく、音量は低下するが有音状態が連続し、しかも相互に関連する信号として取り出すことができる。このように、ステップS18において、デコード結果が0になるようなデータを記憶しておくことにより、オーバーラップした窓関数処理を行って、前後のデータの連続性を確保して滑らかにブロック間をつなぎ、ブロック歪みが軽減されることになる。このため、人間に不快感を与えにくくなる。

－第3の実施形態－

以下、ノイズ除去装置による第3の実施形態について説明する。

【0029】

図10は、本実施形態のエラー検出、およびデコード処理をブロック図で示したものである。エラー検出部14、メモリ15～17、セクタ部19、デコード処理部20からなる。入力データとして1フレームずつ、エラー検出部14とメモリ16に入力される。エラー検出部では、図9におけるステップS11、S13、S14、S15の処理が順次行われる。もしフレーム中にエラーがあると判断した場合には、入力切替信号18によってセクタ部19はメモリ15の内容またはメモリ17の内容をデコード処理部20に出力する。メモリ15にはエラーが発生していない直前のフレームのデータが記憶されており、メモリ17にはデコード結果が0になるフレームパターンが記憶されている。エラー検出部においてフレームにエラーが発生していないと判断した場合には、入力切替信号18によってセクタ部19はメモリ16の値をデコード処理部20に出力する。メモリ16には現在エラー検出を行っているフレームが記憶されているので、フレーム中にエラーが発生していないと判断した場合には、現在のフレームをデコードする通常の処理が行われる。

【 0 0 3 0 】

すなわち、入力切替信号 1 8 によってセクタ部 1 9 はメモリ 1 5 ～ 1 7 のいずれかの値をデコード処理部 2 0 に出力する。

【 0 0 3 1 】

デコード処理部 2 0 では、取得したフレームをデコード処理する。このデコード処理について図 8 を参照して説明する。図 8 の下半分はここで説明するデコード処理をあらわす。周波数サンプル列がメモリ 1 6 で取得したフレームに対応する。デコードするフレームに逆変形離散コサイン変換 I M D C T 処理を行う。

【 0 0 3 2 】

次に、I M D C T の出力であるブロックに窓関数をかける。この窓関数は一種のフィルタとして考えることもでき、各フレームは周波数特性をもつが、窓関数によりこの周波数特性が決定される。A A C では、窓関数としてサイン窓と隣接バンドとの分離度の良いガイザー窓の 2 種類が定義されており、本実施形態ではいずれの窓関数をも選択することが可能である。

【 0 0 3 3 】

窓関数処理は一つの I M D C T の出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバーラップさせて窓関数処理を行う。図 8 においてブロックは隣接する前後のブロックのブロックサイズの半分ずつをオーバーラップさせて窓関数処理を行っている。窓関数処理を行った出力ブロックの隣接するブロックのブロックサイズの半分ずつを加算して、信号を再生する。

【 0 0 3 4 】

したがって、フレームでエラーが検出された場合、メモリ 1 5 の内容（エラーが発生していない直前のフレームデータ）または、メモリ 1 7 の内容（デコード処理の結果が“0”となるフレームデータ）をデコード処理するが、いずれの場合もエラーが発生したフレームをデコード処理した結果が無音となることはなく、しかも前後のフレームから出力音を再生するので、前後のデータの連続性が確保されており、エラー検出時における出力音をより聞きやすい音に改善することができる。

【 0 0 3 5 】

また、本出願によるノイズ除去方法はノイズ除去装置によって実施される図 7 および図 8 において明示されている。

【 0 0 3 6 】

さらに、上記図 7 および図 9 に示すフローチャートに対応するプログラムをフレキシブルディスクまたはハードディスク等の媒体に記録しておき、或いはインターネット等のネットワークを用いて配信しておき、これをマイクロコンピュータ等により読み出して実行することにより、当該マイクロコンピュータ等をシステムコントローラとして動作させることも可能である。

【 0 0 3 7 】

本出願にかかわる実施形態は、M D C T を使用している M P 3、A C - 3、M P E G 4 あるいは A T R A C などにも適応することができる。

【図面の簡単な説明】

【図 1】

A D T S フレームの構造を示す図。

【図 2】

R a w _ d a t a _ b l o c k の構造を示す図。

【図 3】

I D の種類を示す図。

【図 4】

従来の A A C デコーダのエラー検出方法のフローチャート。

【図 5】

第 1 の従来技術を示す概略図。

【図 6】

第 2 の従来技術を示す概略図。

【図 7】

第 1 の実施形態のエラー検出方法フローチャート。

【図 8】

エンコード処理およびデコード処理の手順。

【図 9】

第 2 の実施形態のエラー検出方法のフローチャート。

【図 1 0】

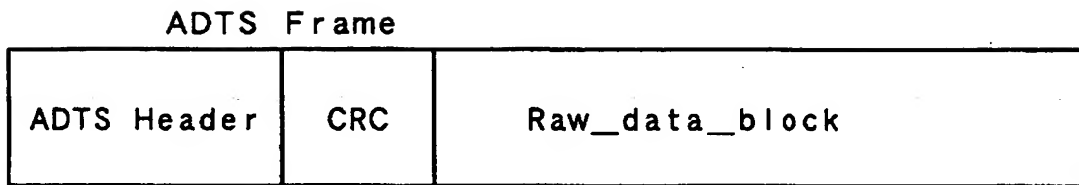
第 3 の実施形態のエラー検出およびデコード処理のブロック図。

【符号の説明】

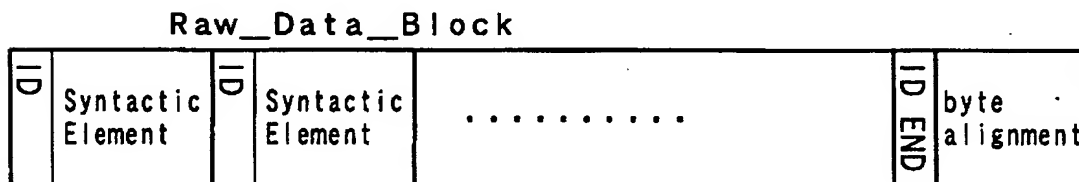
- 6、13：時間サンプル列
- 9：周波数サンプル列
- 10：IMDCT
- 11：窓関数
- 12：加算
- 14：エラー検出部
- 15、16、17：メモリ
- 18：入力切替信号
- 19：セレクタ部
- 20：デコード処理部
- 21、22、23：フレーム
- 21A、22A、23A：ブロック
- 24：システムコントローラ

【書類名】 図面

【図 1】



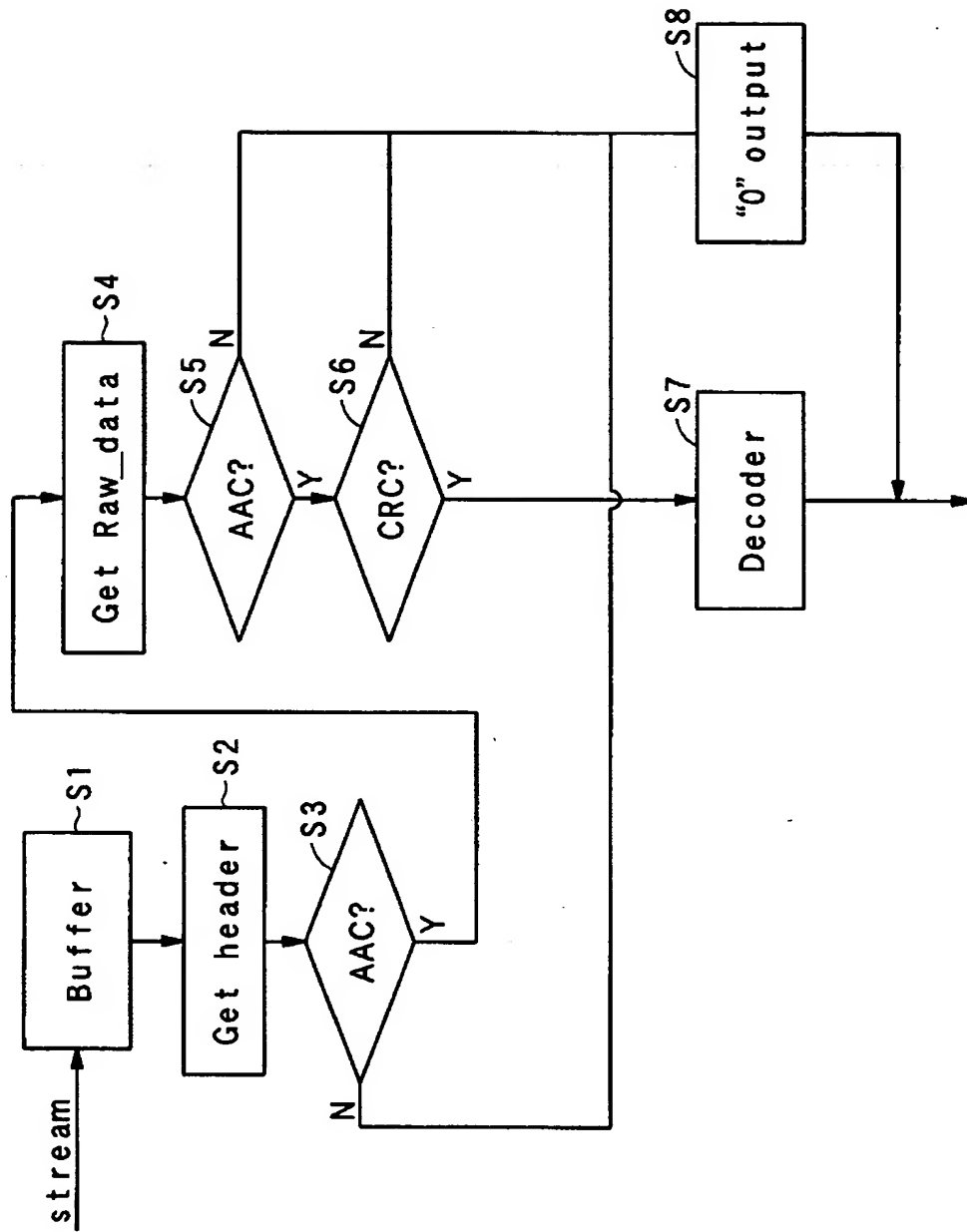
【図 2】



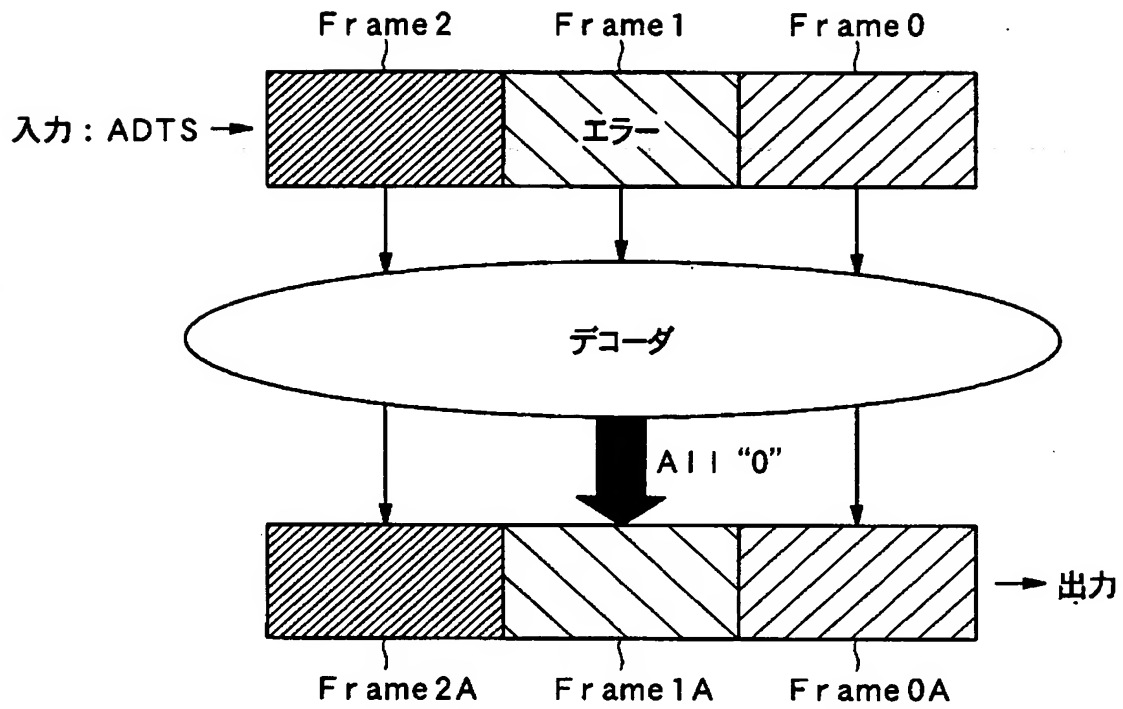
【図 3】

Syntactic Element	ID name	Encoding	Abbreviation
single_channel_element	ID_SCE	0x0	SCE
channel_pair_element	ID_CPE	0x1	CPE
coupling_channel_element	ID_CCE	0x2	CCE
life_channel_element	ID_LFE	0x3	LFE
Data_stream_element	ID_DSE	0x4	DSE
program_config_element	ID_PCE	0x5	PCE
fill_element	ID_FIL	0x6	FIL
Terminator	ID_END	0x7	TERM

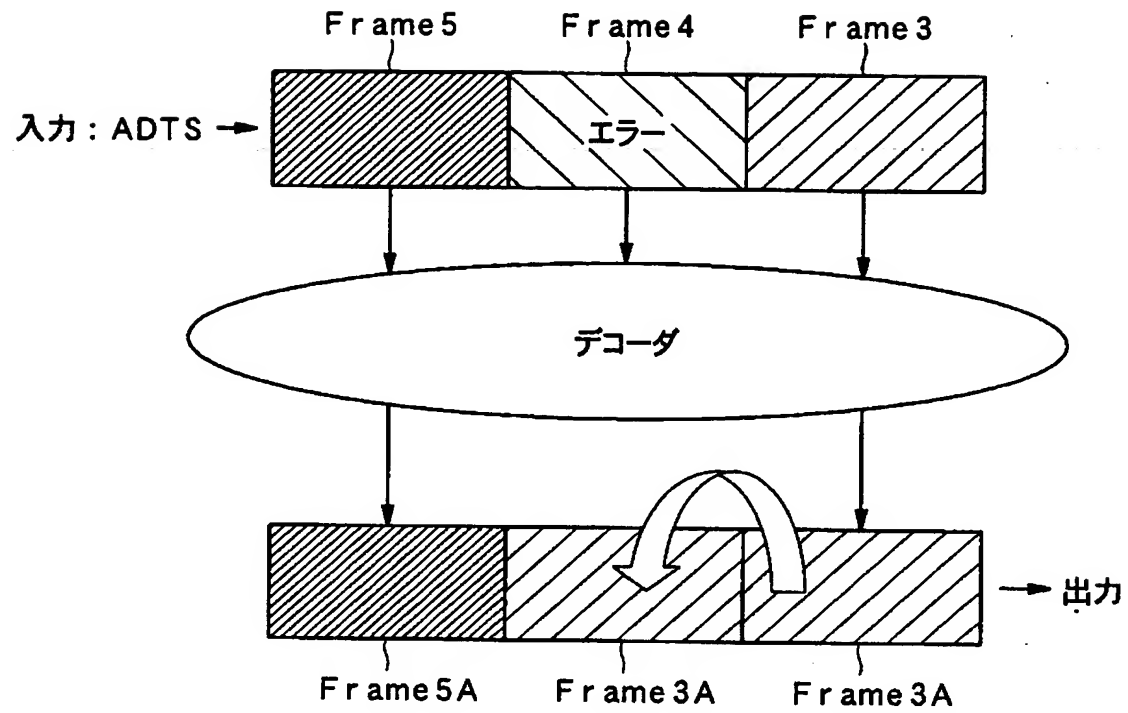
【図 4】



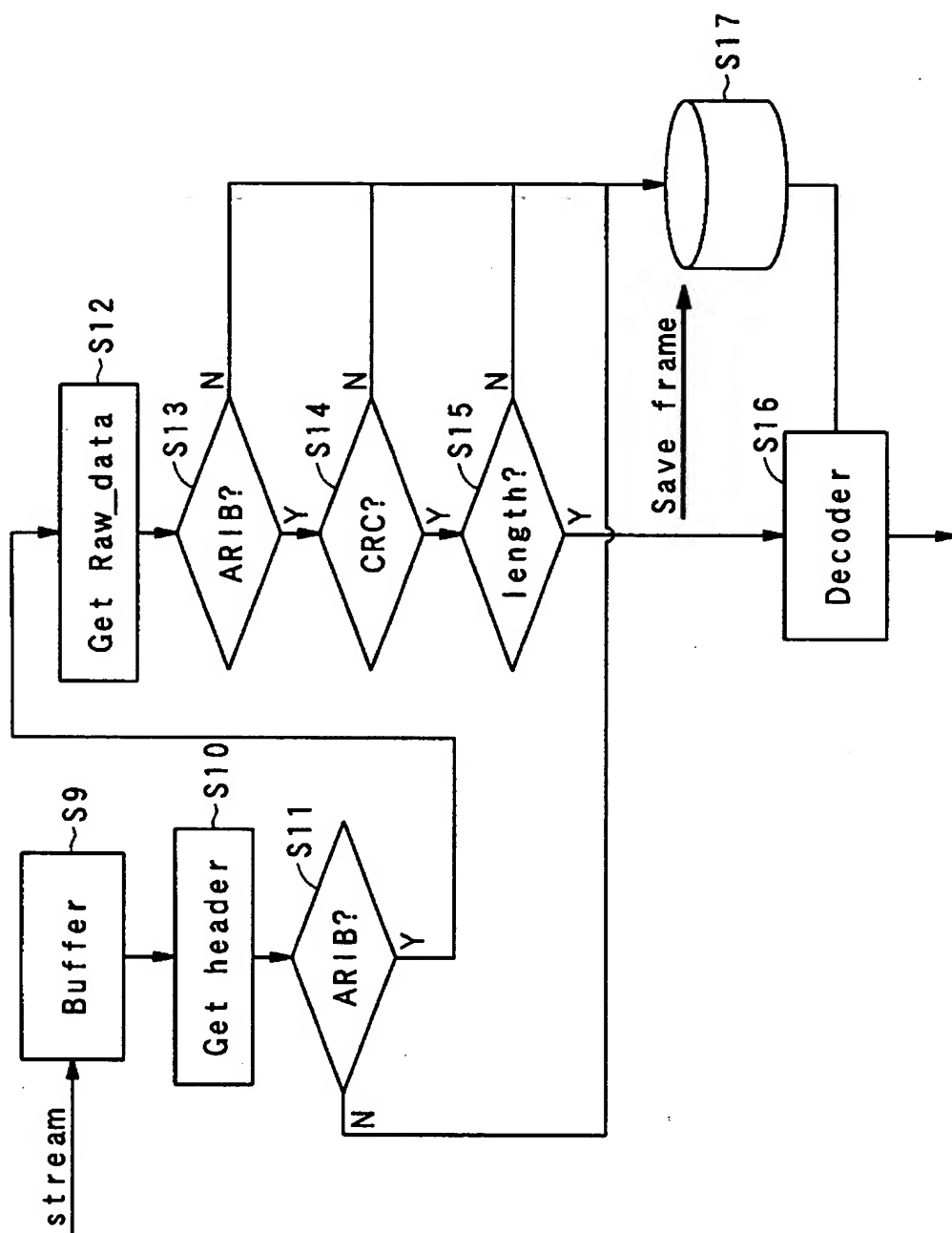
【図 5】



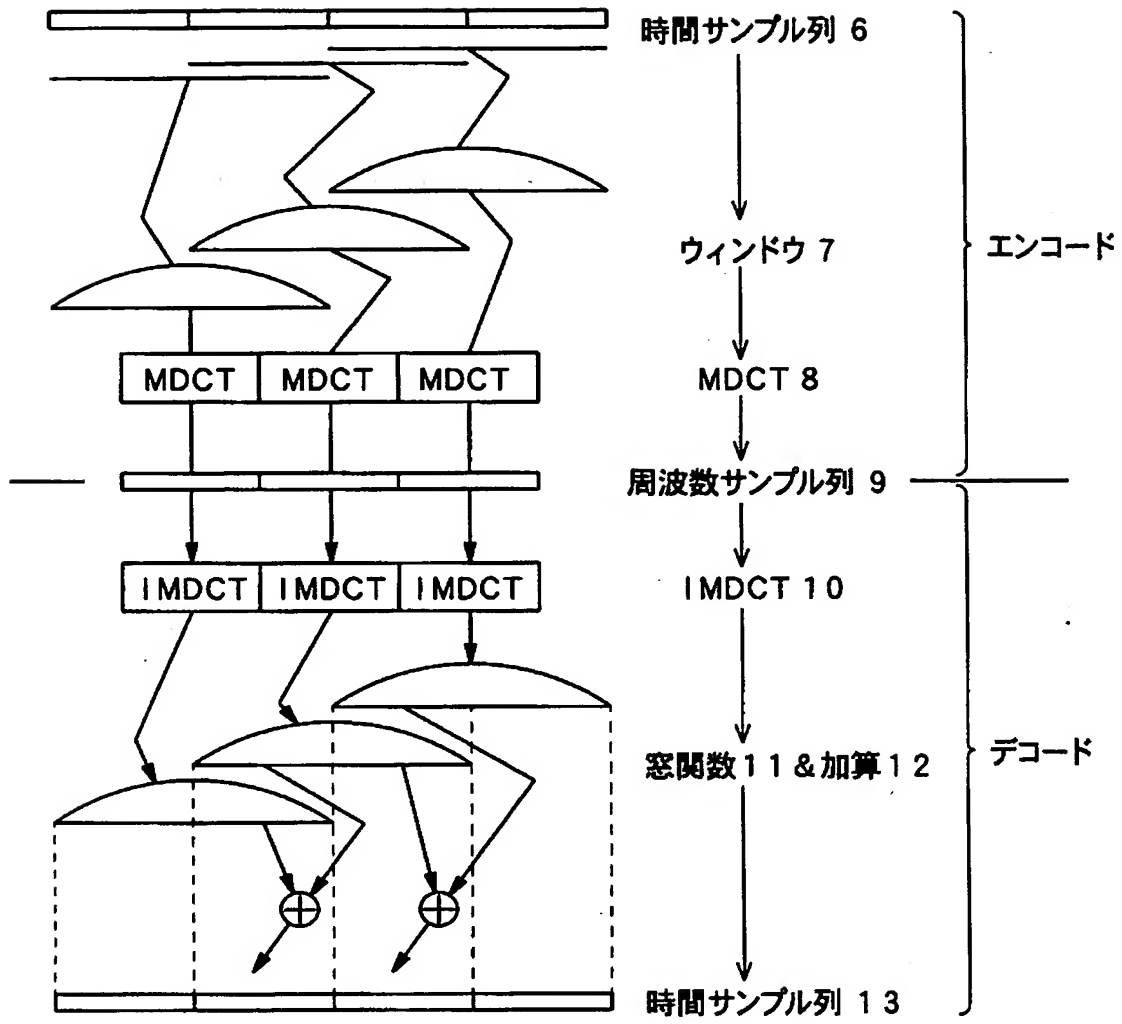
【図 6】



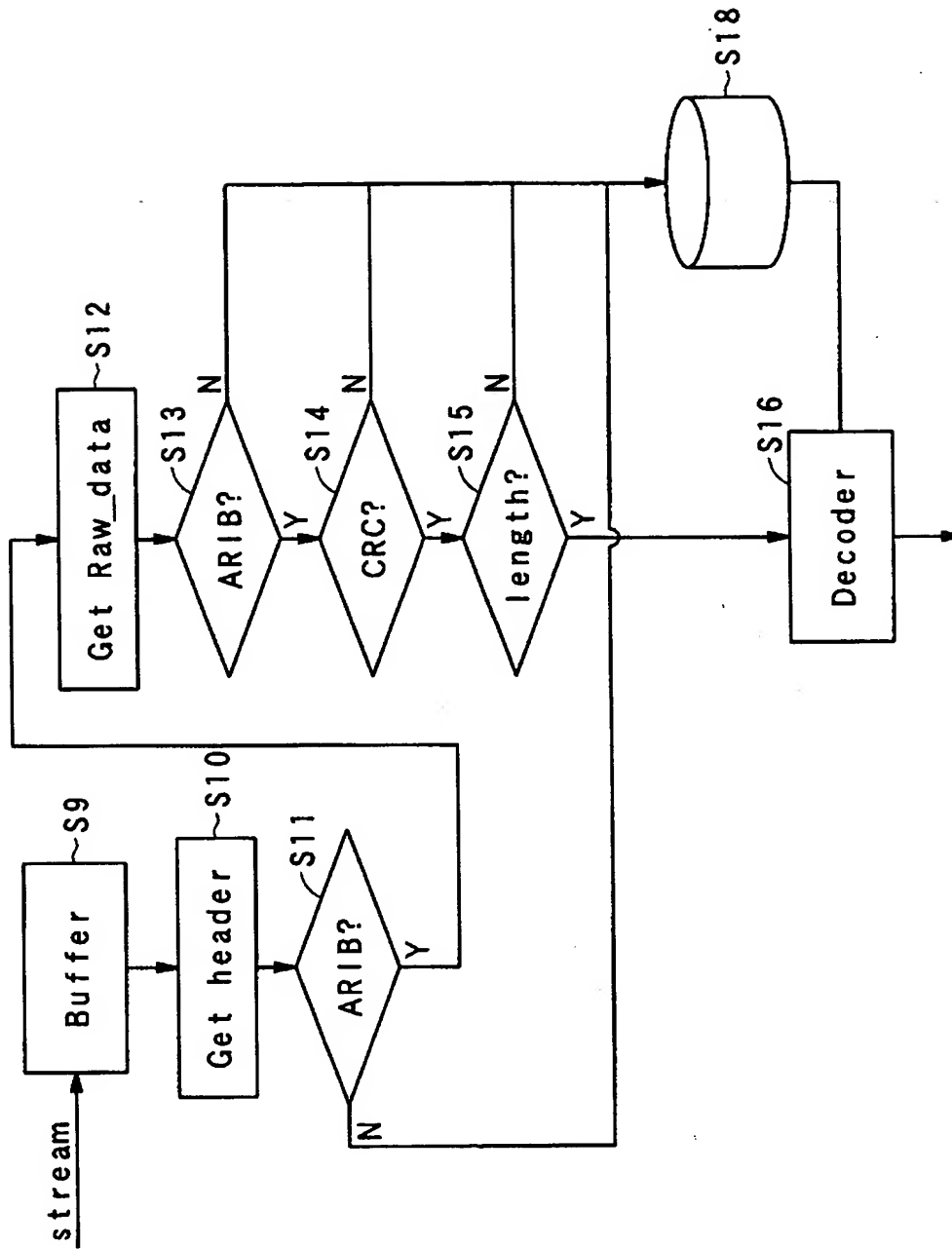
【図 7】



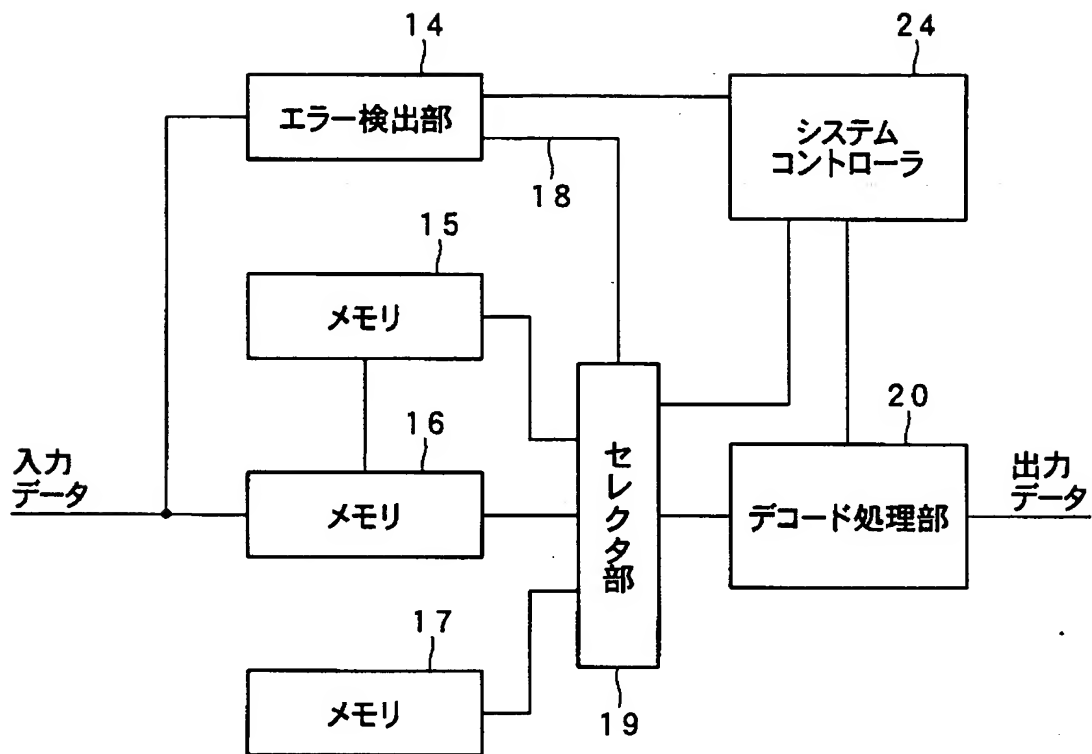
【図 8】



【図 9】



【図 10】



【書類名】 要約書

【要約】

【課題】 CRCによるエラー検出よりもさらにエラー検出能力を高め、かつ、エラー検出時における出力音をより聞きやすい音に改善することを課題の一例としている。

【解決手段】 上記の課題を解決するために、請求項1に記載の発明は、符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする。

【選択図】 図9

出 願 人 履 歴 情 報

識別番号 [000005016]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都目黒区目黒1丁目4番1号

氏 名 パイオニア株式会社